FIGURE 1 (Prior Art)

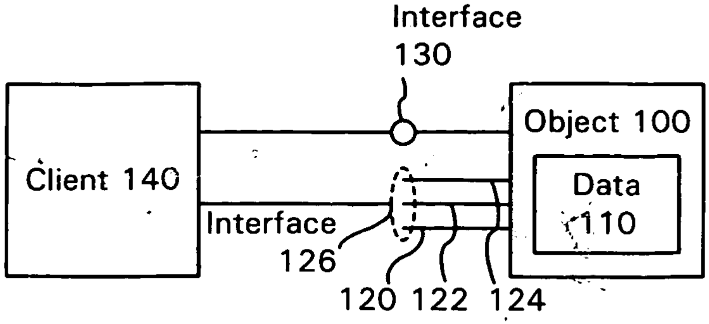


FIGURE 3 (Prior Art)

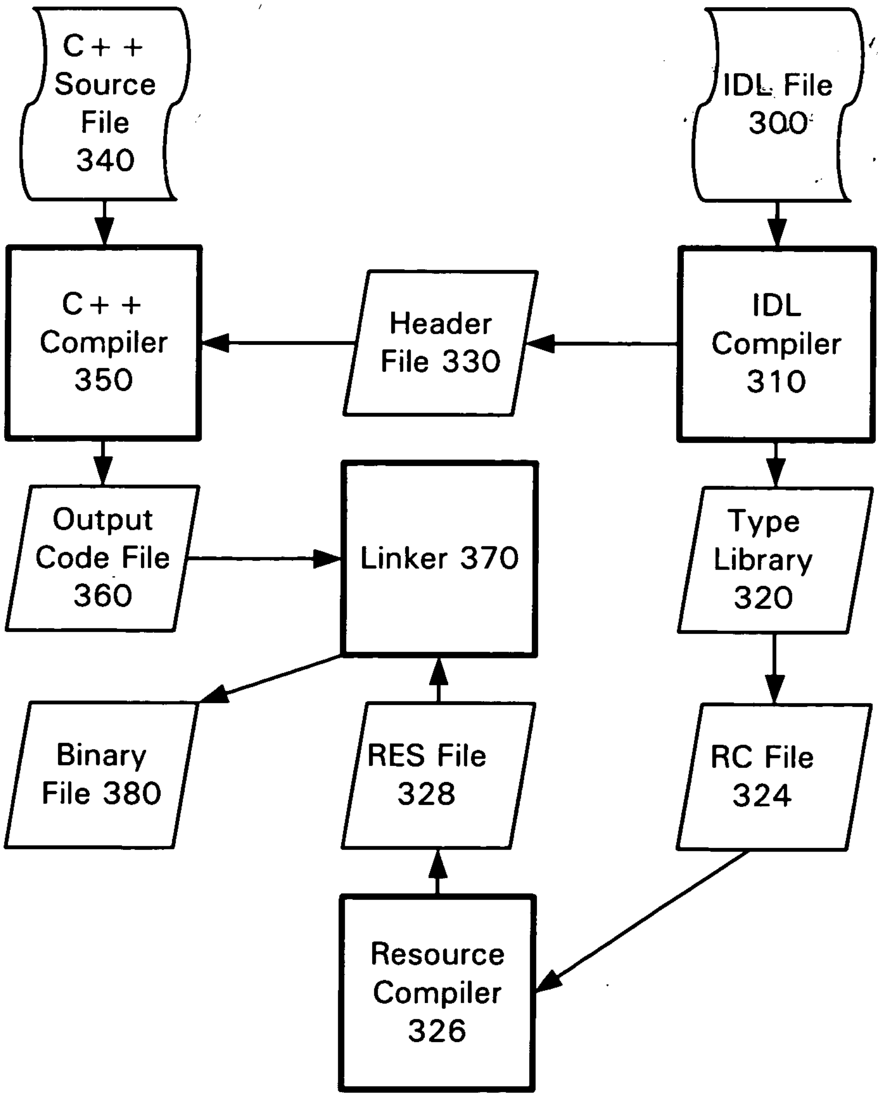


FIGURE 2a (Prior Art)

200

```
import "docobj.idl";

enum E {
   e = 17,
};
struct S {
   int i, j;
};

[version(1.0), helpfile("test.res"), helpcontext(12), uuid(eed3644c-8488-3ecd-ba97-147db3cdb499) ]
library MyLib
{
   importlib("stdole2.tlb");
   importlib("olepro32.dll");

   [uuid(1AECC9B3-2104-3723-98B8-7CC54722C7DD), helpstring("interface ITest")]
   dispinterface ITest {
   properties:
   methods:
      [id(34)] void Grade([in] enum E , [out,retval] char *);
      [id(18)] HRESULT Score([in] struct S *a, [in] float b, [in] VARIANT c);
   };
```

210

To Figure 2b

FIGURE 2b (Prior Art)

To Figure 2a

200

230

```
[object, uuid(1DAD4027-2BA5-34F1-AD39-76A637E6579E), helpstring("interface ITest2")]
interface ITest2 : IUnknown {
   void  __cdecl  Display();
   [propput] void  StudentID([in] int );
   [propget] void  StudentID([out,retval] int *);
   HRESULT  Hours([in] int , [in] float );
};

[uuid(12341234-1234-1234-1234-123412341234), version(1.0), helpstring("interface CTest")]
coclass CTest {
   dispinterface ITest;
   interface ITest2;
};
}
```

250

FIGURE 4

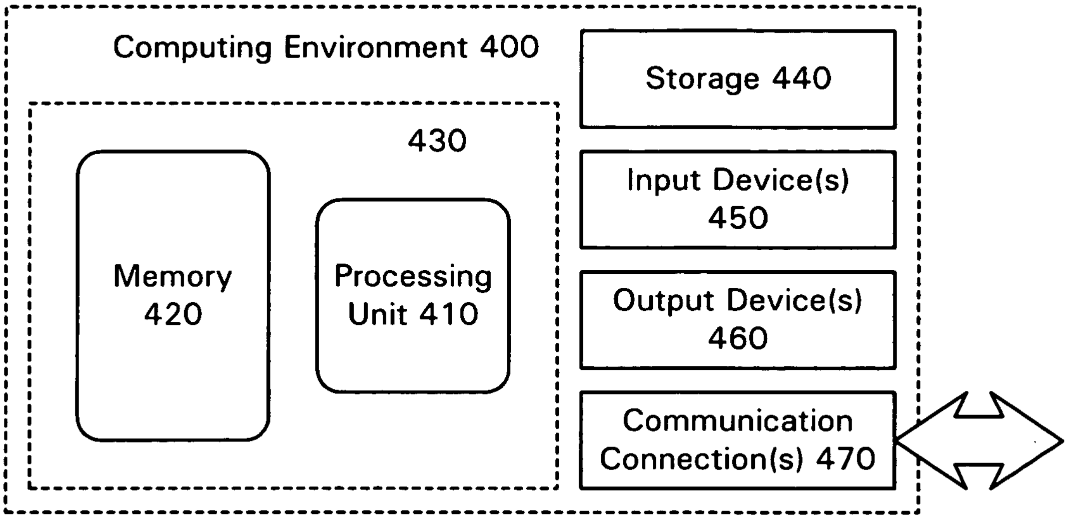
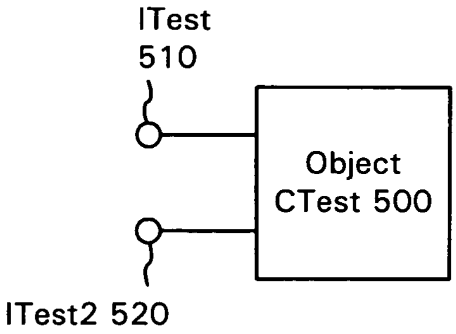


FIGURE 5



600

```
#define _ATL_ATTRIBUTES 1
#include "atlbase.h"
#include "atlcom.h"
extern "C" int printf(const char*, ...);

// IDL library block and overall project settings
[project(type=dll, name=MyLib, helpfile="test.res", helpcontext=12)];

[export] enum E {
   e = 17
};
[export] struct S {
   int i, j;
};

[dispinterface, helpstring("interface ITest")] __interface ITest : IDispatch {
      [id(34)] void Grade([in] E, [out, retval] char*);
      [id(18)] HRESULT Score([in]S̀* a, [in]float b, [in]VARIANT c);
};

[object, library_block, helpstring("interface ITest2")] __interface ITest2 {
      void __cdecl Display(void);
      [propput] void StudentID([in] int);
      [propget] void StudentID([out,retval] int*);
      HRESULT Hours([in]int, [in]float);
};
```

610

620

630

640

To Figure 6b

FIGURE 6a

To Figure 6a

600

650

660

```
[coclass, progid(CTest.17), helpstring("interface CTest"), uuid(12341234-1234-1234-1234-123412341234)]
struct CTest : ITest, ITest2 {
   void Grade(E e, char* pc) {
      printf("CTest::Grade(e=%d)\n", (int) e);
      *pc = 'A';
   }
   HRESULT Score(S* a, float b, VARIANT c) {
      printf("CTest::Score(a=%p,b=%f,c=%d)\n", a, b, c.lVal);
      return S_OK;
   }
   void Display() {
      printf("CTest::Display()\n");
   }
   void put_StudentID(int i) {
      printf("CTest::put_StudentID(i=%d)\n", i);
   }
   void get_StudentID(int* ) {
      printf("CTest::get_StudentID()\n");
   }
   HRESULT Hours(int a, float b) {
      printf("CTest::Hours(a=%d,b=%f)\n", a, b);
      return S_OK;
   }
};
```

FIGURE 6b

FIGURE 7

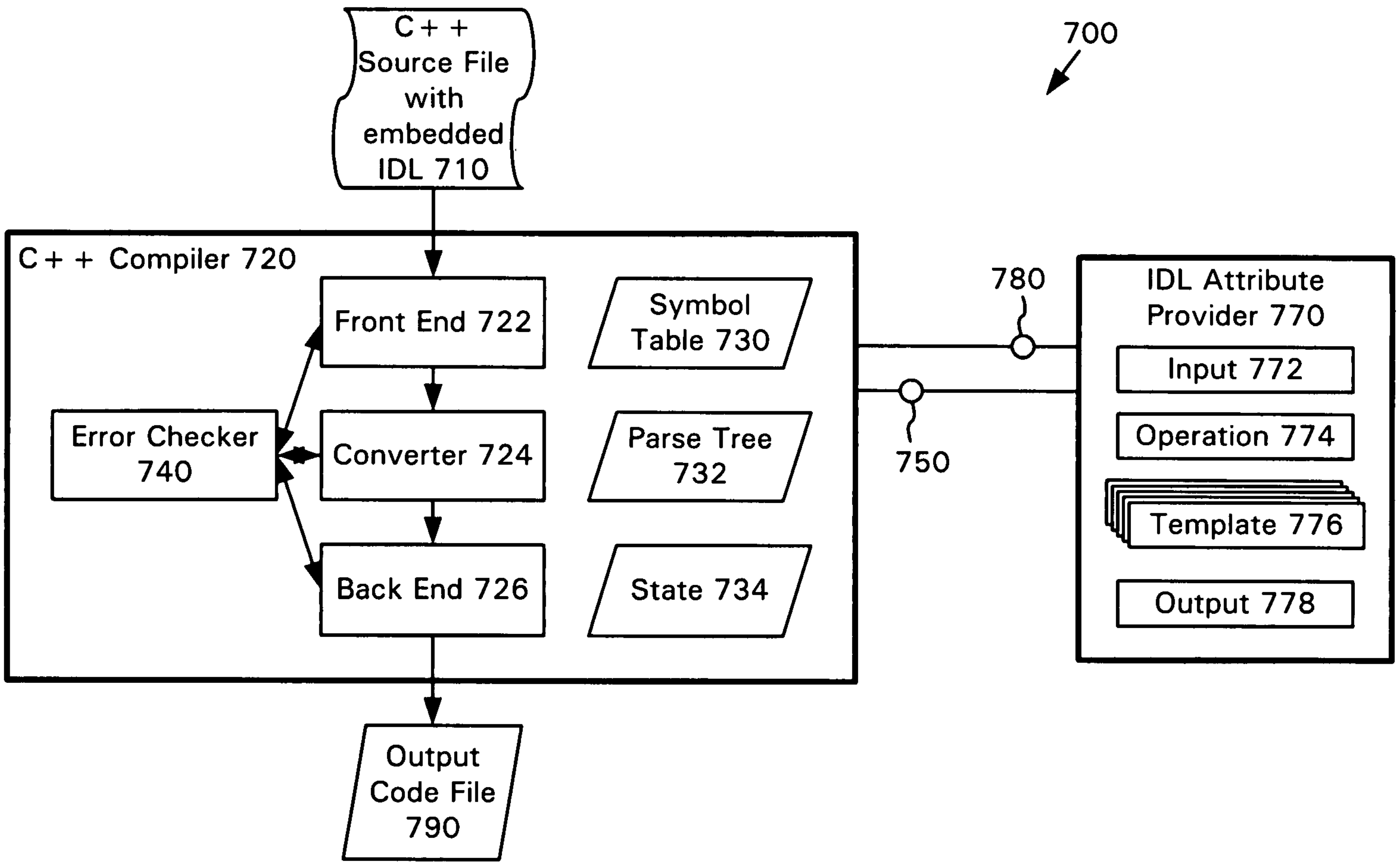
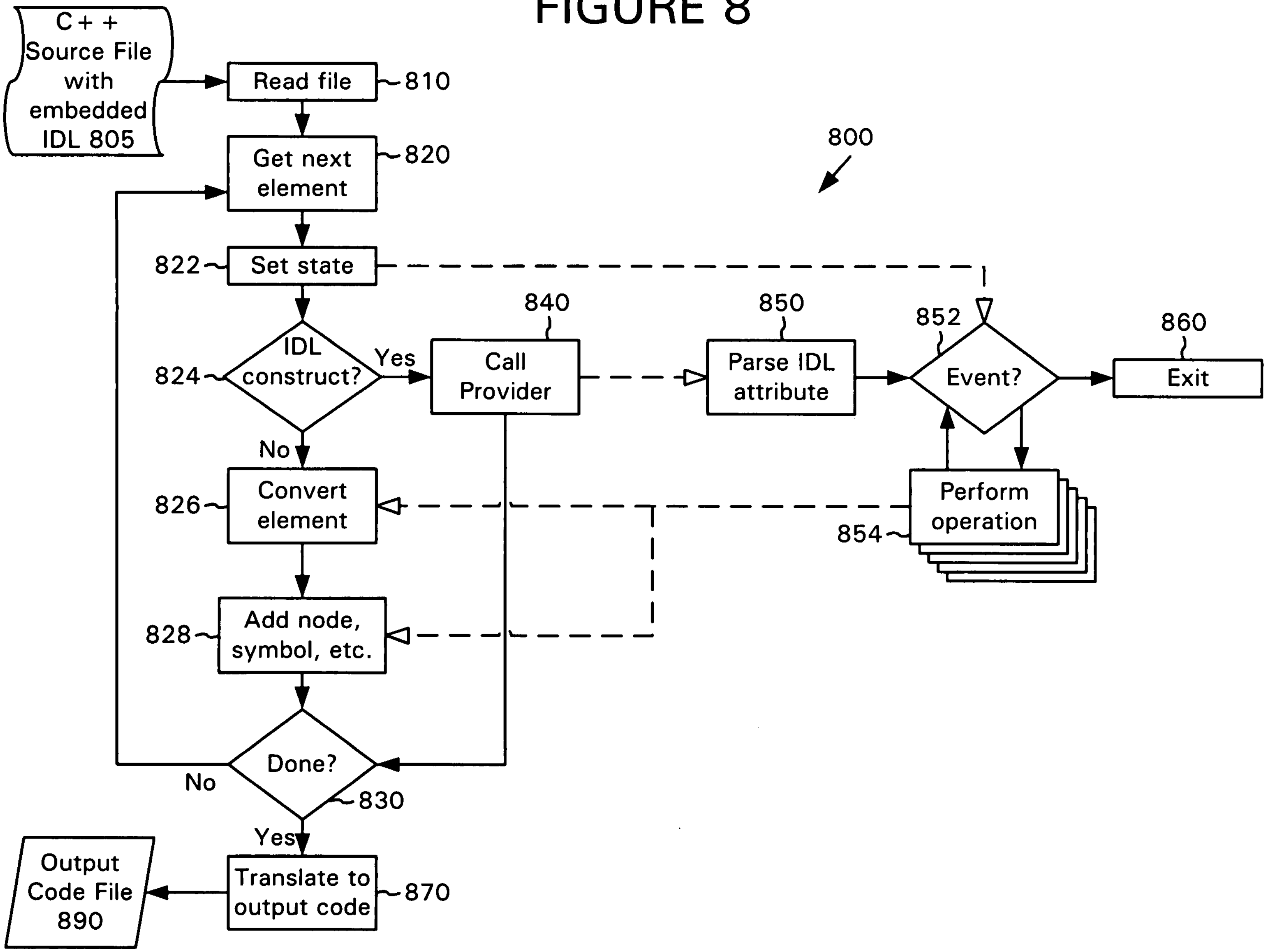

700
C + +
Source File
with
embedded
IDL 710
C + + Compiler 720
Front End 722
Error Checker
740
Converter 724
Back End 726
Symbol
Table 730
Parse Tree
732
State 734
780
750
IDL Attribute
Provider 770
Input 772
Operation 774
Template 776
Output 778
Output
Code File
790

FIGURE 8


C + + Source File with embedded IDL 805
Read file
810
Get next element
820
800
822
Set state
824
IDL construct?
Yes
No
Call Provider
840
Parse IDL attribute
850
852
Event?
860
Exit
826
Convert element
Perform operation
854
828
Add node, symbol, etc.
Done?
No
830
Yes
Translate to output code
870
Output Code File 890

900

```
[coclass, progid(CTest.17), helpstring("interface CTest"), uuid(12341234-1234-1234-1234-123412341234)]
struct CTest : ITest, ITest2 ,
   /* + + + Added Baseclass */ public CComCoClass<CTest, &__uuidof(CTest)>,
   /* + + + Added Baseclass */ public CComObjectRootEx<CComSingleThreadModel>,
   /* + + + Added Baseclass */ public IProvideClassInfoImpl<&__uuidof(CTest)>
{
   void Grade(E e, char* pc) {
      printf("CTest::Grade(e=%d)\n", e);
      *pc = 'A';
   }
   HRESULT Score(S* a, float b, VARIANT c) {
      printf("CTest::Score(a=%p,b=%f,c=%d)\n", a, b, c.iVal);
      return S_OK;
   }
   void Display() {
      printf("CTest::Display()\n");
   }
   void put_StudentID(int i) {
      printf("CTest::put_StudentID(i=%d)\n", i);
   }
   void get_StudentID(int* ) {
      printf("CTest::get_StudentID()\n");
   }
   HRESULT Hours(int a, float b) {
      printf("CTest::Hours(a=%d,b=%f)\n", a, b);
      return S_OK;
   }
```

FIGURE 9a

900

```
//+ + + Start Injected Code
virtual HRESULT STDMETHODCALLTYPE Invoke(
            /* [in] */ DISPID dispIdMember,
            /* [in] */ REFIID riid,
            /* [in] */ LCID lcid,
            /* [in] */ WORD wFlags,
            /* [out][in] */ DISPPARAMS *pDispParams,
            /* [out] */ VARIANT *pVarResult,
            /* [out] */ EXCEPINFO *pExcepInfo,
            /* [out] */ UINT *puArgErr)
{
    HRESULT hr = S_OK;
    if (pDispParams = = 0) {
        return DISP_E_BADVARTYPE;
    }
    if (pVarResult != 0) {
        VariantInit(pVarResult);
    }
    switch (dispIdMember) {
    case 18:
        {
            S* i1 = (S*) V_RECORD(&pDispParams->rgvarg[2]);
            float i2 = V_R4(&pDispParams->rgvarg[1]);
            VARIANT i3 = pDispParams->rgvarg[0];
```

FIGURE 9b

900

```
            hr = Score(i1, i2, i3);
            if (pVarResult != 0) {
                V_VT(pVarResult) = VT_ERROR;
                V_ERROR(pVarResult) = hr;
            }
            break;
        }
    case 34:
        {
            E i1 = (E) V_I4(&pDispParams->rgvarg[1]);
            char* i2 = (char*) V_I1REF(&pDispParams->rgvarg[0]);
            Grade(i1, i2);
            if (pVarResult != 0) {
                V_VT(pVarResult) = VT_UI1 | VT_BYREF;
                V_I1REF(pVarResult) = (char*) i2;
            }
            break;
        }
    default:
        return DISP_E_MEMBERNOTFOUND;
    }
    return hr;
}
```

FIGURE 9c

900

```
virtual HRESULT STDMETHODCALLTYPE GetIDsOfNames(
            /* [in] */ REFIID riid,
            /* [size_is][in] */ LPOLESTR *rgszNames,
            /* [in] */ UINT cNames,
            /* [in] */ LCID lcid,
            /* [size_is][out] */ DISPID *rgDispId)
{
    static LPOLESTR names[] = { L"Grade", L"Score" };
    static DISPID dids[] = { 34, 18 };
    for (unsigned int i = 0; i < cNames; + +i) {
        int fFoundIt = 0;
        for (unsigned int j = 0; j < sizeof(names)/sizeof(LPOLESTR); + +j) {
            if (lstrcmpW(rgszNames[i], names[j]) = = 0) {
                fFoundIt = 1;
                rgDispId[i] = dids[j];
            }
        }
        if (fFoundIt = = 0) {
            return DISP_E_UNKNOWNNAME;
        }
    }
    return S_OK;
}
```

FIGURE 9d

900

```
HRESULT TypeInfoHelper(REFIID iidDisp, LCID /*lcid*/, ITypeInfo** ppTypeInfo)
{
   if (ppTypeInfo == NULL) {
      return E_POINTER;
   }
   *ppTypeInfo = NULL;
   TCHAR szModule1[_MAX_PATH];
   ::GetModuleFileName(_pModule->GetModuleInstance(), szModule1, _MAX_PATH);
   USES_CONVERSION;
   CComPtr<ITypeLib> spTypeLib;
   HRESULT hr = LoadTypeLib(T2OLE(szModule1), &spTypeLib);
   if (SUCCEEDED(hr)) {
      CComPtr<ITypeInfo> spTypeInfo;
      hr = spTypeLib->GetTypeInfoOfGuid(iidDisp, &spTypeInfo);
      if (SUCCEEDED(hr)) {
         *ppTypeInfo = spTypeInfo.Detach();
      }
   }
   return hr;
}
```

FIGURE 9e

900

```
    virtual HRESULT STDMETHODCALLTYPE GetTypeInfoCount(unsigned int*  pctinfo)
    {
        if (pctinfo = = NULL) {
            return E_POINTER;
        }
        CComPtr<ITypeInfo> spTypeInfo;
        *pctinfo =
                (SUCCEEDED(TypeInfoHelper(__uuidof(ITest), 0, &spTypeInfo))) ? 1 : 0;
        return S_OK;
    }
    virtual HRESULT STDMETHODCALLTYPE GetTypeInfo(unsigned int iTInfo, LCID lcid, ITypeInfo** ppTInfo)
    {
        if (iTInfo != 0) {
            return DISP_E_BADINDEX;
        }
        return TypeInfoHelper(__uuidof(ITest), lcid, ppTInfo);
    }
//+ + + End Injected Code
......................................
};
```

FIGURE 9f

FIGURE 10a

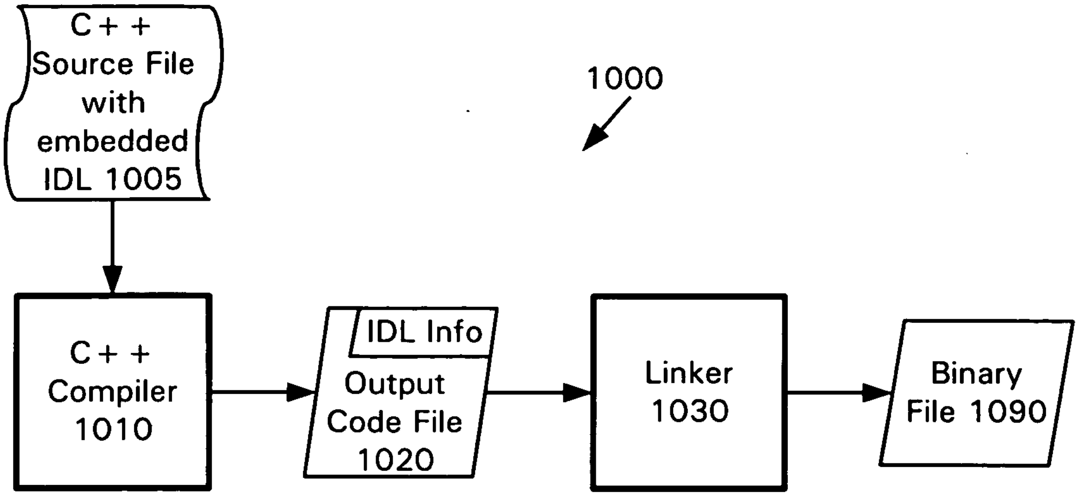
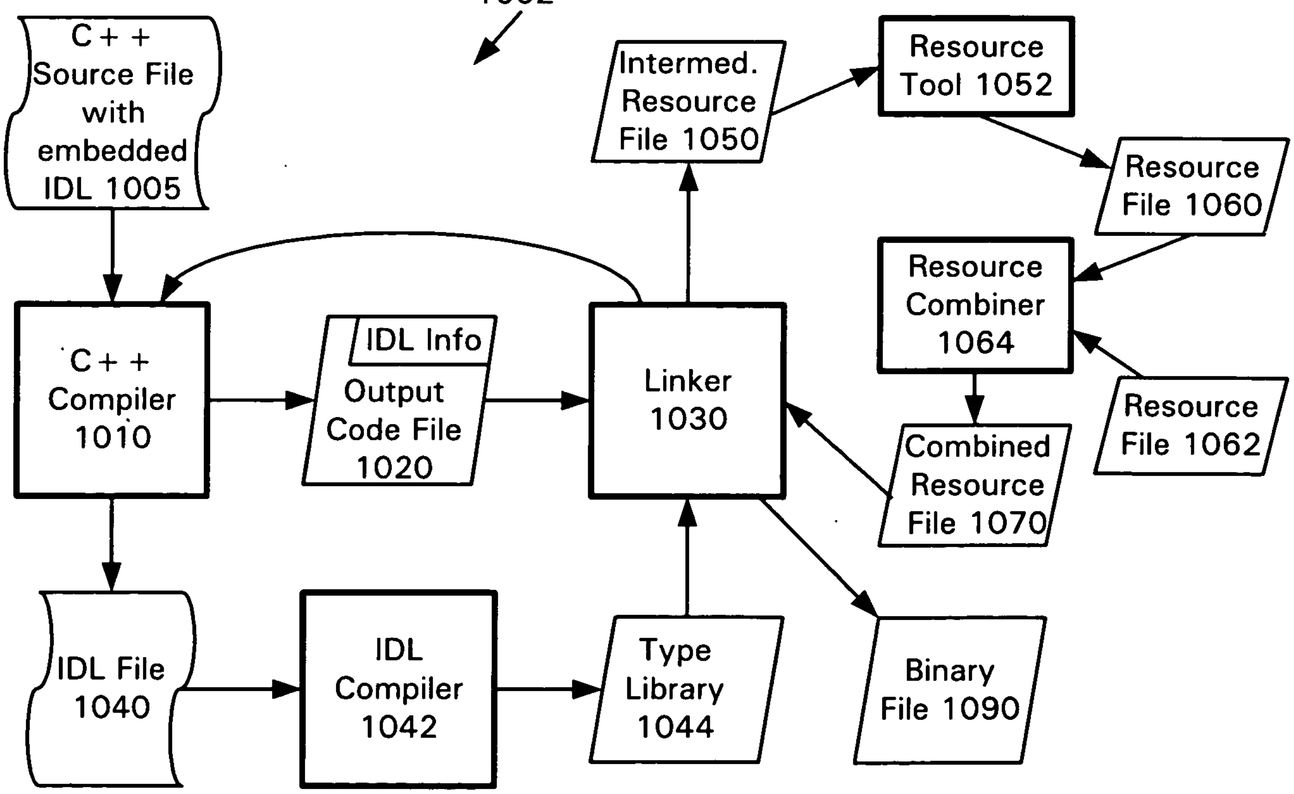


FIGURE 10b


1002
C++ Source File with embedded IDL 1005
Intermed. Resource File 1050
Resource Tool 1052
Resource File 1060
Resource Combiner 1064
C++ Compiler 1010
IDL Info
Output Code File 1020
Linker 1030
Resource File 1062
Combined Resource File 1070
IDL File 1040
IDL Compiler 1042
Type Library 1044
Binary File 1090